PHIL 7001: Fundamentals of AI, Data, and Algorithms

Week 9
Non-Linear Learning

Boris Babic,
HKU 100 Associate Professor of Data Science, Law and Philosophy

Today

Learning goals

- Introduction to Random Forests and Support Vector Machines (SVM)
- Understanding Random Forests: Essential concepts and usefulness
- SVM's approach to classification and separating hyperplanes
- Comparison of Logistic Regression, Random Forests, and SVM

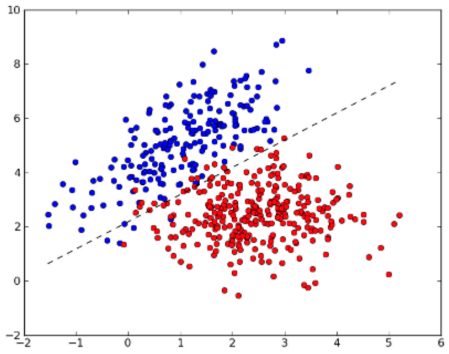
Last week

Review of last week

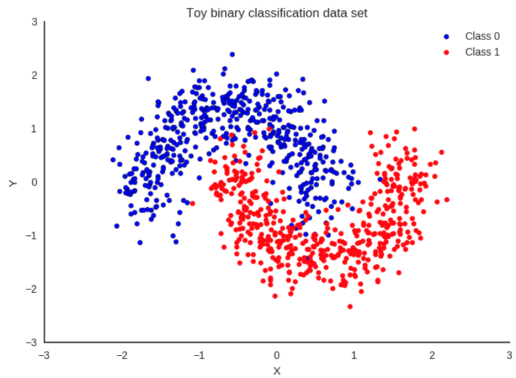
- Intro to classification
- Logistic regression
- Making probabilistic predictions
- Using logistic regression for classification
- Evaluating accuracy of classifiers

Logistic Regression

- Recall that logistic regression is a linear model that works well when the relationship between features and the target is relatively simple and linear.
- It's very efficient when the dataset has features that are linearly separable.
- However, it has limitations when dealing with complex and nonlinear data patterns.
  - Assumes linearity between dependent and independent variables.
  - Non-linear problems can't be solved with logistic regression since it has a linear decision surface
  - Fails to capture complex relationships.
- We may need models that are more complicated



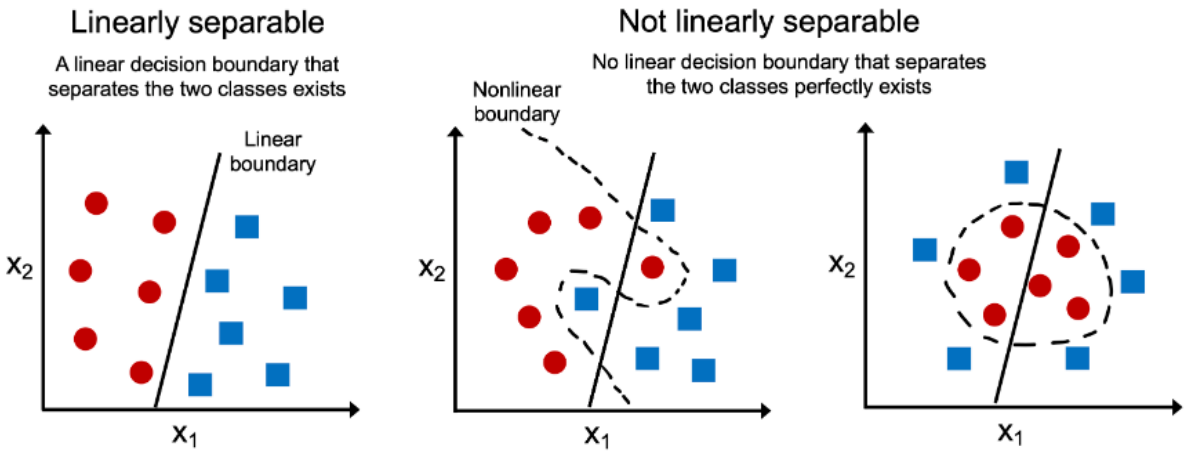
Linearly separable



Nonlinearly separable

Nonlinear Models

- In cases where the data is nonlinear, using more complicated models becomes necessary to accurately capture the underlying patterns.
- Nonlinear models, like Random Forests and SVM, create more flexible decision boundaries, resulting in better classification performance.
- Nonlinear models have the advantage of capturing intricate relationships.
  - Better classification accuracy, especially when the data contains complex patterns that cannot be adequately represented by a linear model.
- However, nonlinearity comes with its own challenges, primarily the risk of overfitting.
  - Higher capacity to fit noise in the data, then poor generalization to new, unseen data.

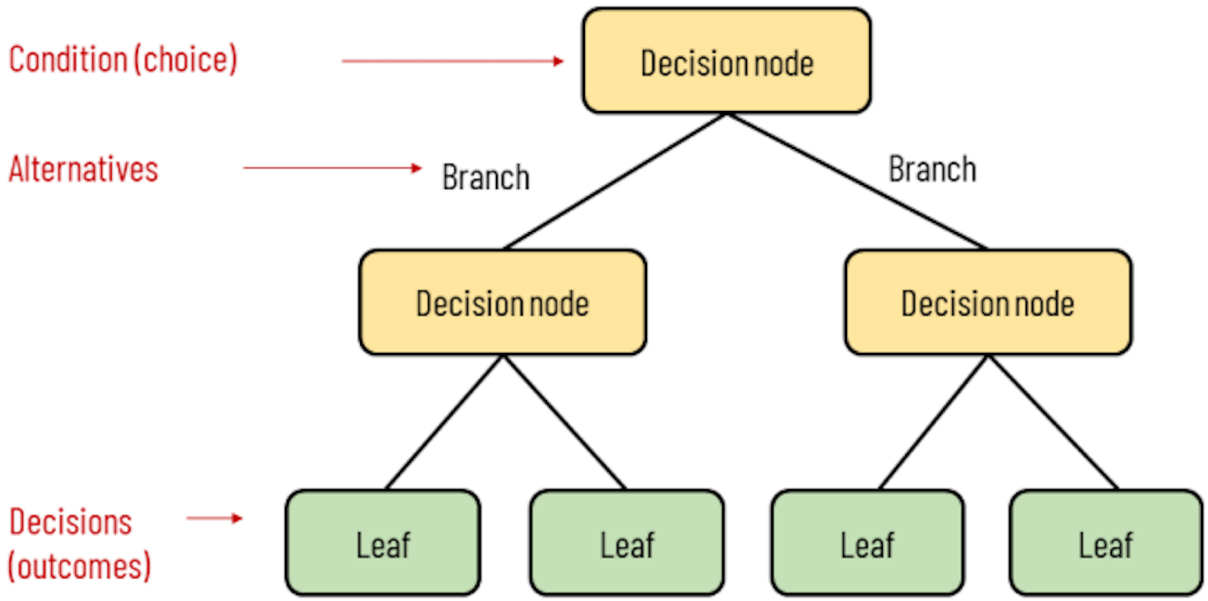


Transition: From Logistic Regression to RF and SVM

- As we tackle more intricate classification challenges, the limitations of logistic regression become evident.
- Complex real-world data often defies linear relationships. We need models capable of adapting to nonlinear patterns.
- Building on Logistic Regression:
  - Random Forests (RF):
    - Exploits the foundation of logistic regression while branching into an ensemble approach.
  - Support Vector Machines (SVM):
    - Originates from linear regression but focuses on maximizing separation between classes.
    - Adapts logistic regression's concept of hyperplanes for classification.
- While these models offer enhanced capabilities, overfitting remains a concern.
- Model tuning and validation remain crucial to strike a balance between complexity and generalization.

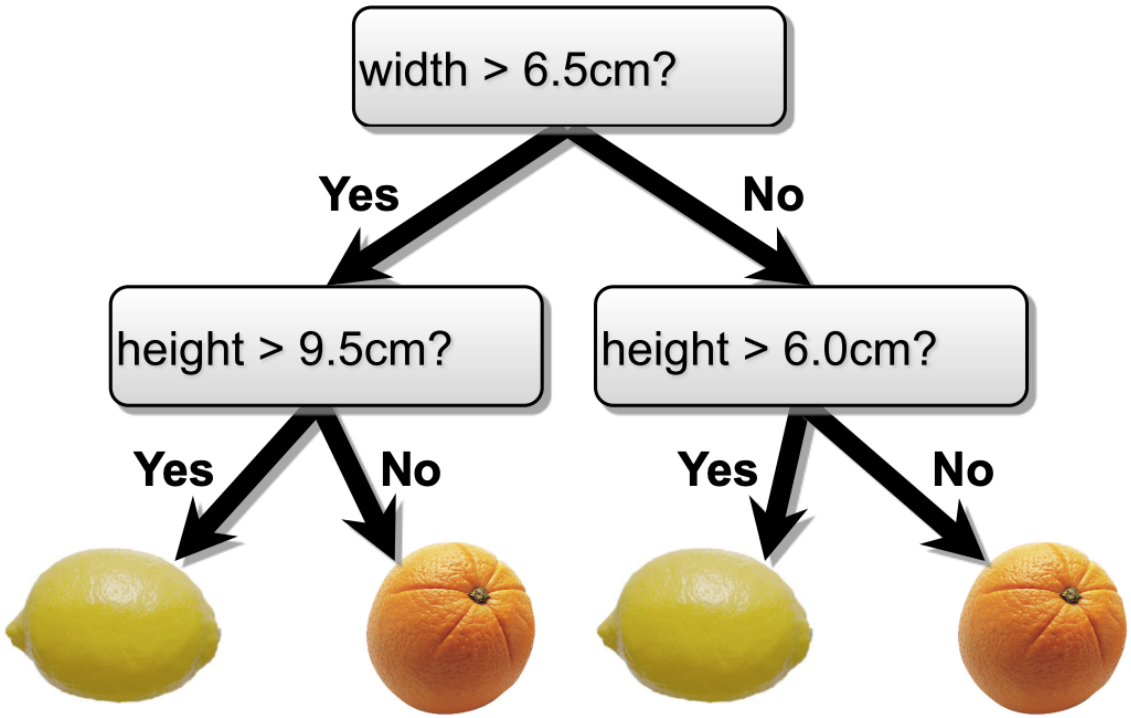
Decision Trees

- A decision tree is a simple model for supervised classification. It is used for classifying a single discrete target feature.
- It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.
- Imagine a flowchart-like structure where each internal node represents a decision based on a feature, each branch represents an outcome of that decision, and each leaf node represents a class label.



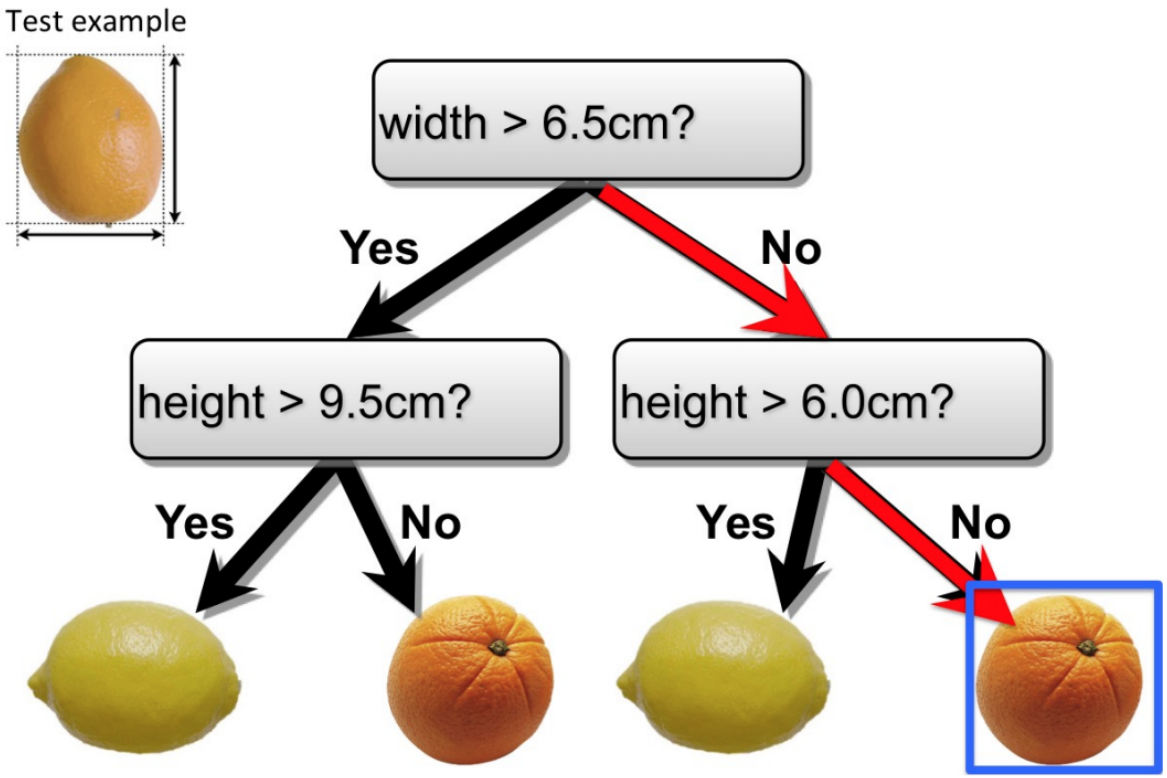
Decision Trees

- Make predictions by splitting on attributes according to a tree structure.



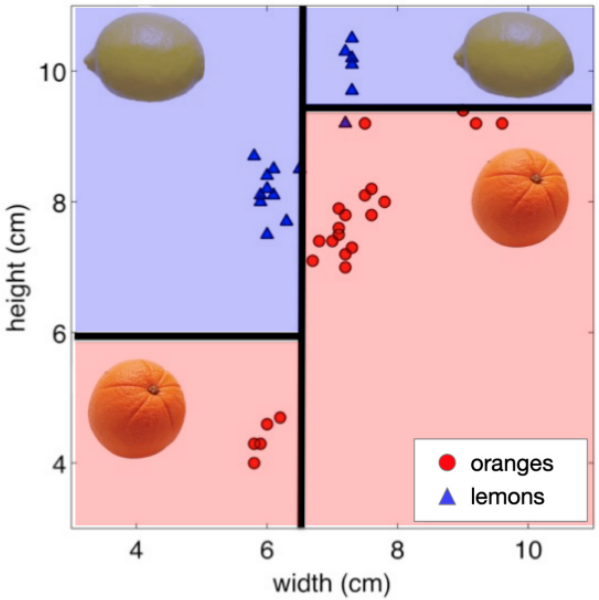
Decision Trees

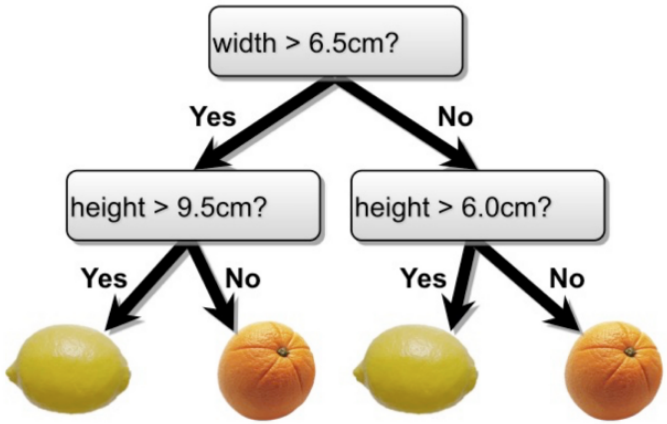
- Make predictions by splitting on attributes according to a tree structure.
- Given that we have an object with 4.0cm width and 5.5cm height, what will be our prediction?



Decision Trees

- For continuous attributes, we partition the range by checking whether that attribute is greater than or less than some threshold.
- Decision boundary is made up of axis-aligned planes.



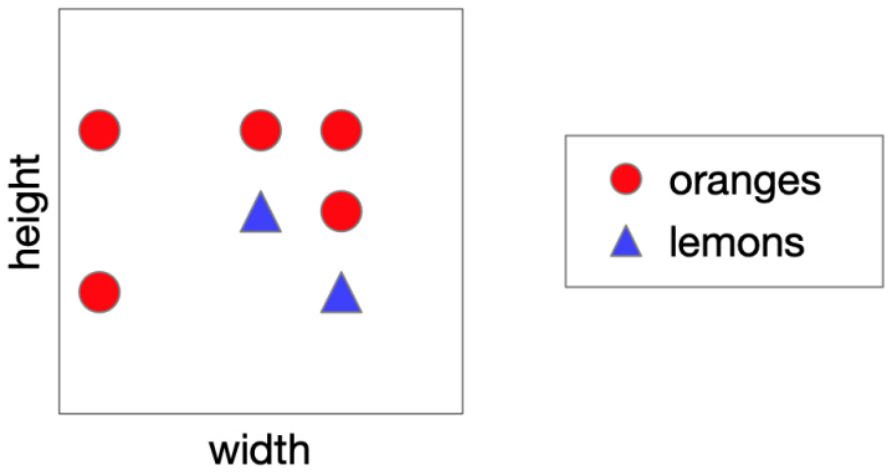


Decision Trees

- The process of splitting is repeated in a top-down, recursive manner until all, or the majority of objects have been classified under specific class labels.
- For any training set we can construct a decision tree that has exactly one leaf for every training point, but it probably won't generalize.
  - it can often lead to overfitting.
- As a result, decision trees have preference for small trees.
- **But how do we determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned?**

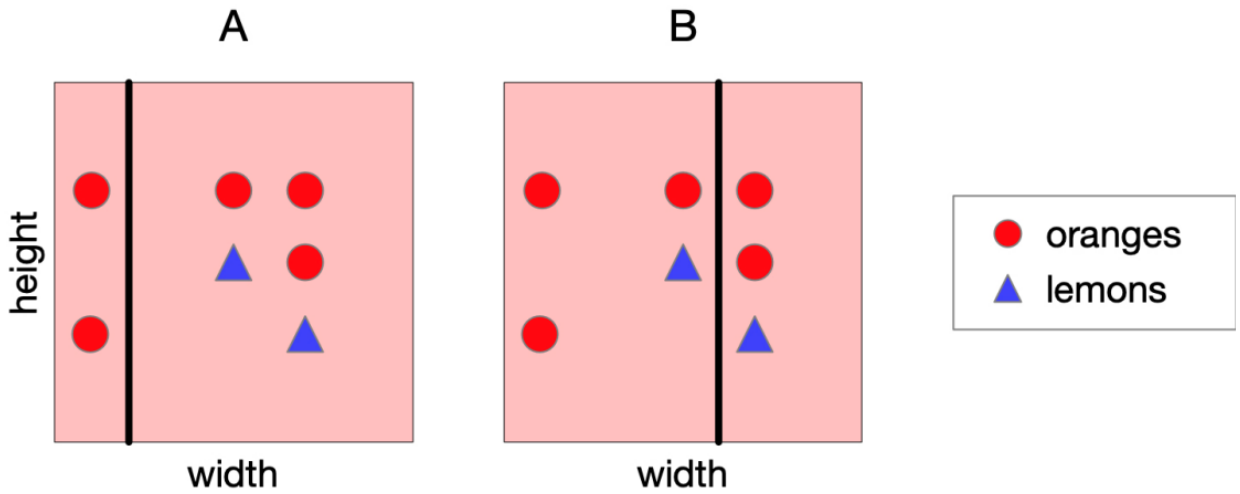
Decision Trees - Entropy

- Consider the following data. Let's split on width.



Decision Trees - Entropy

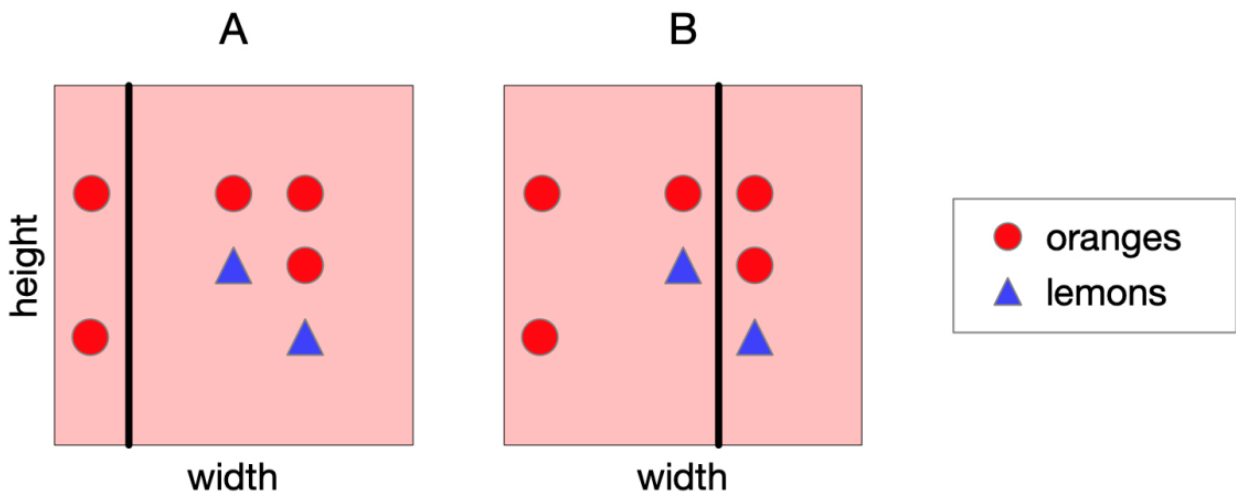
- Recall: classify by majority



- A and B have the same misclassification rate, so which is the best split?

Decision Trees - Entropy

- Recall: classify by majority



- A and B have the same misclassification rate, so which is the best split?
- A feels like a better split, because the left-hand region is very certain about whether the fruit is an orange.
- Can we quantify this?

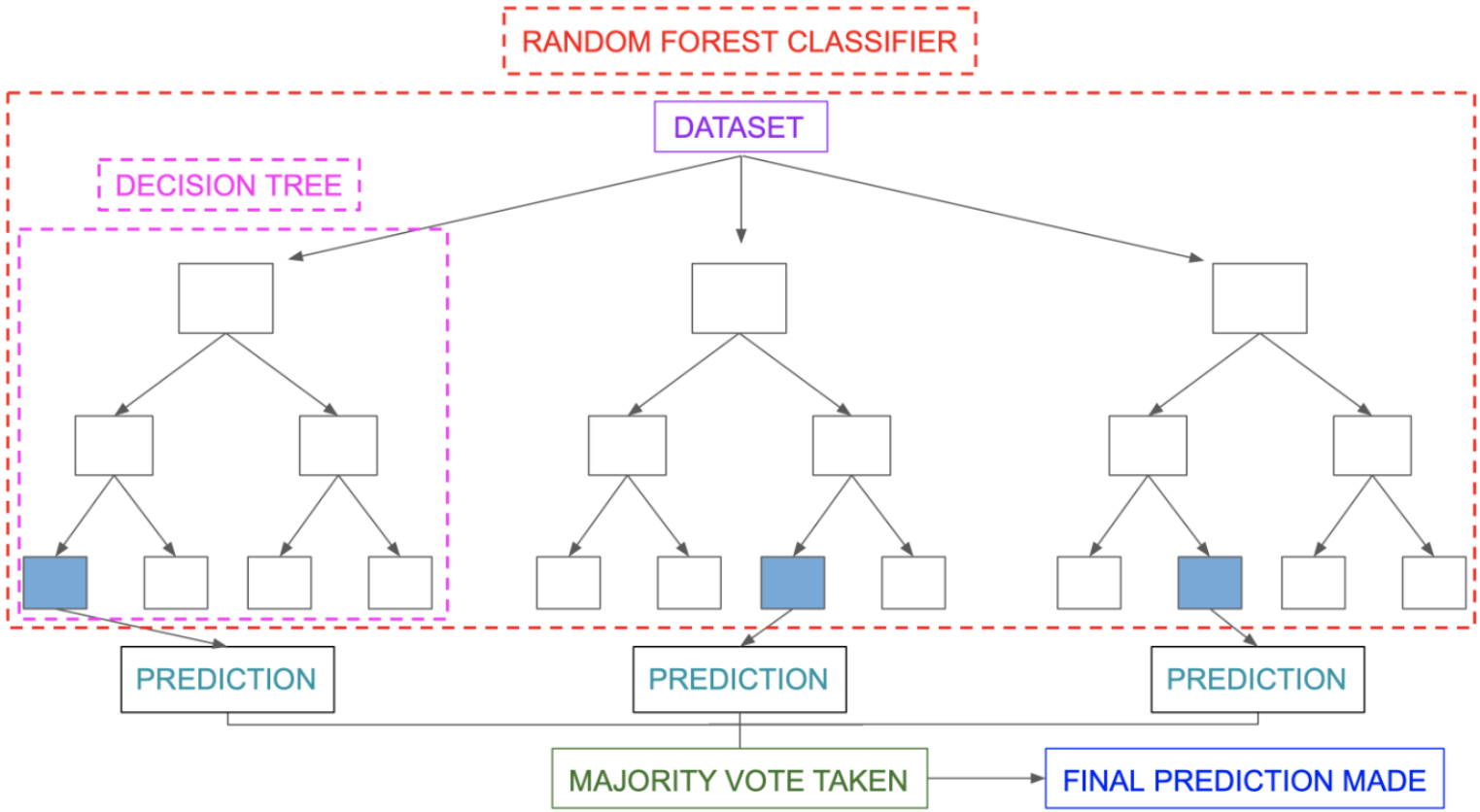
Decision Trees - Entropy

- How can we quantify uncertainty in prediction for a given leaf node?
  - If all examples in leaf have same class: good, low uncertainty
  - If each class has same amount of examples in leaf: bad, high uncertainty
- Use counts at leaves to define probability distributions; use a probabilistic notion of uncertainty to decide splits.
- The **entropy** is a number that quantifies the uncertainty inherent in its possible outcomes.
  - $-plog_2(p) - (1-p)log_2(1-p)$
- The mathematical definition of entropy may seem arbitrary, but the overall idea is that it is used to quantify the uncertainty associated with a set of data points, with higher entropy scores indicating higher uncertainty level.

Random Forests

- Random Forests take the concept of a decision tree and elevate it to a whole new level.
- Instead of relying on a single decision tree, a Random Forest consists of an ensemble of multiple decision trees.
- Each tree is trained on a subset of the data and makes its own classification decisions.
- The final prediction is then determined by aggregating the predictions of all individual trees.

Random Forests



Random Forests

- Why generalize this notion with Random Forests?
- Well, think about it. While a single decision tree can easily become overly complex and prone to overfitting, an ensemble of diverse trees can work together to create a more robust and accurate classification model.
- This diversification reduces the risk of overfitting and yields a more stable and reliable prediction.

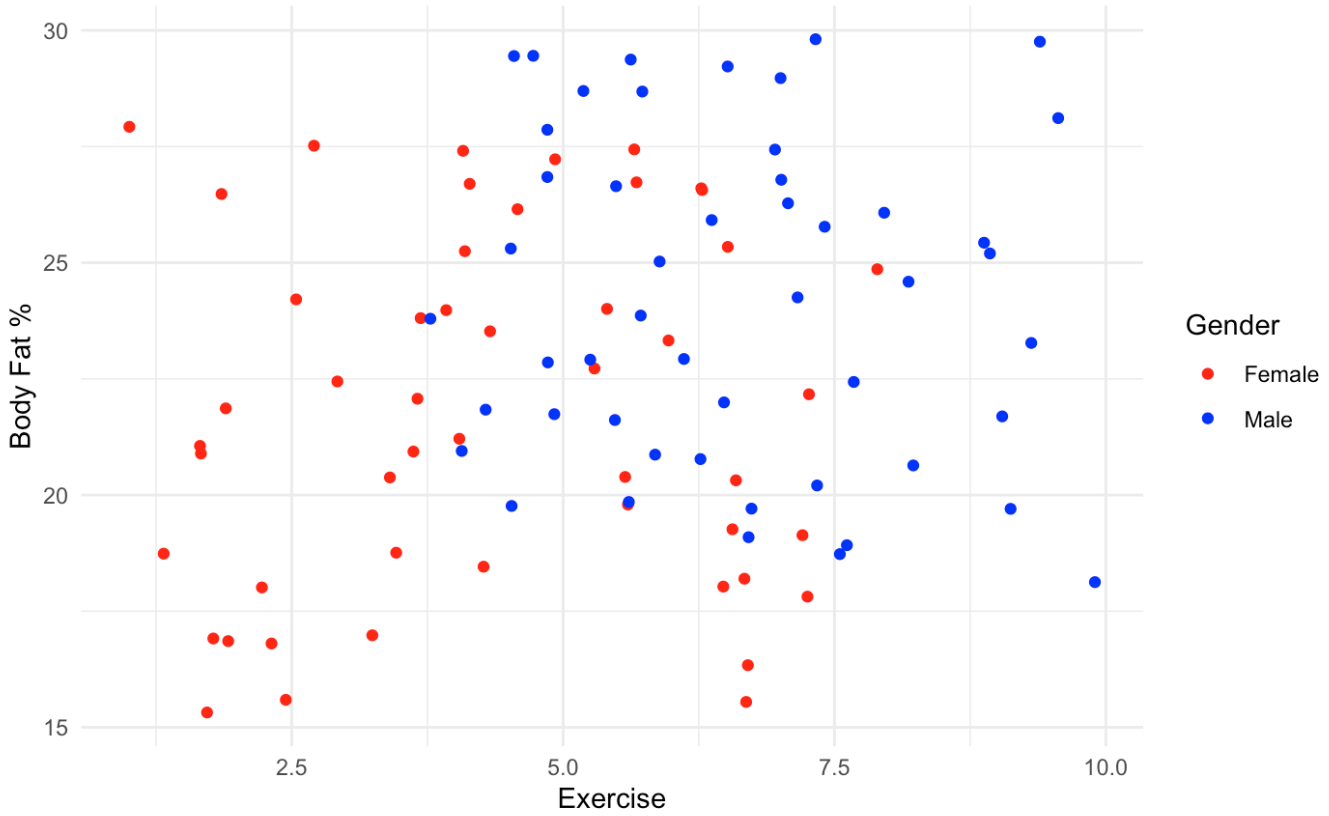
Random Forests - Example

- Remember the scenario we worked on in our logistic regression lecture.
- Let's apply a Random Forests classifier to create a classification boundary to predict the gender (male or female) based on body fat percentage and exercise hours per week.
  - We're dealing with binary classification (two classes).
  - Male and female are our class labels.
- Again, we will divide the data into training set(70%) and testing set(30%).

```
# Glimpse the dataset
glimpse(data)
```

```
## Rows: 100
## Columns: 3
## $ exercise <dbl> 1.320818, 4.095401, 6.592474, 1.853295, 4.926636, 2.445720, 1…
## $ body_fat <dbl> 18.73851, 25.24797, 20.31670, 26.47923, 27.22607, 15.59223, 2…
## $ gender   <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
```

Random Forests - Example



Random Forests - Example

- In R, **randomForest()** is the function we can use to build the Random Forest model.
- 'gender exercise + body_fat' is the formula we provide. This formula specifies the relationship we want to model.
- In this case, we're trying to predict the 'gender' variable based on the 'exercise' and 'body_fat' variables.
- 'ntree = 500' specifies that we want to create 500 decision trees in our Random Forest. More trees usually lead to better generalization and accuracy.

```
# Train the Random Forest classifier
rf_model <- randomForest(gender ~ exercise + body_fat, data = data_train, ntree = 500)
```

Random Forests - Example

- We are using the test data for model evaluation.
- The '**predict()**' function is used to get predictions from the trained random forests classifier.
- Recall that a confusion matrix is a table that allows us to understand the performance of a classification model.

```
# Make predictions on the test data
predictions <- predict(rf_model, data_test)

# Create a confusion matrix
conf_matrix <- table(predictions, data_test$gender)
print(conf_matrix)
```

```
##
## predictions 0 1
##           0 8 7
##           1 7 8
```

Random Forests - Example

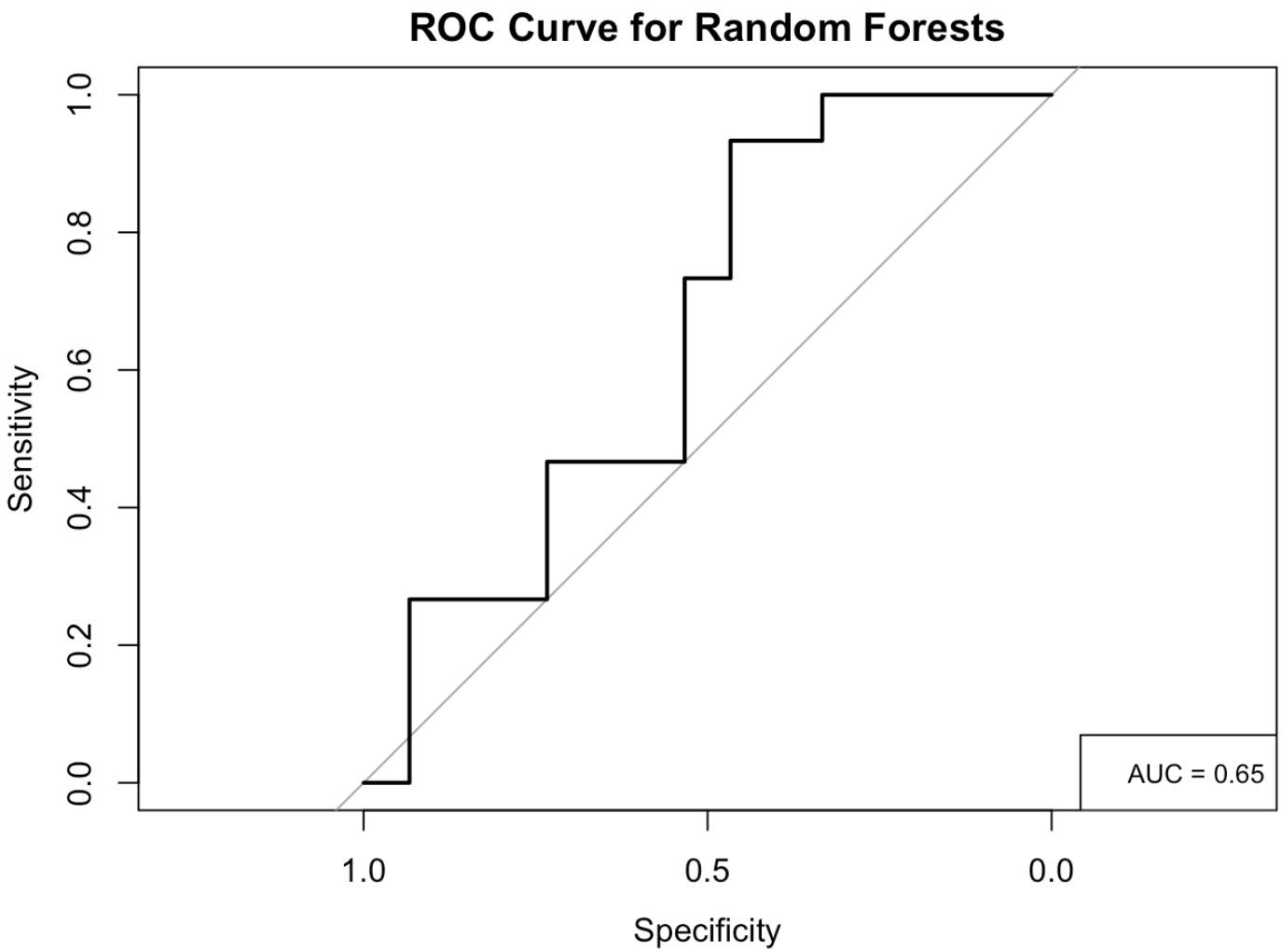
	Predicted 0	Predicted 1
Actual 0	True Negative	False Positive
Actual 1	False Negative	True Positive

Table: Confusion Matrix

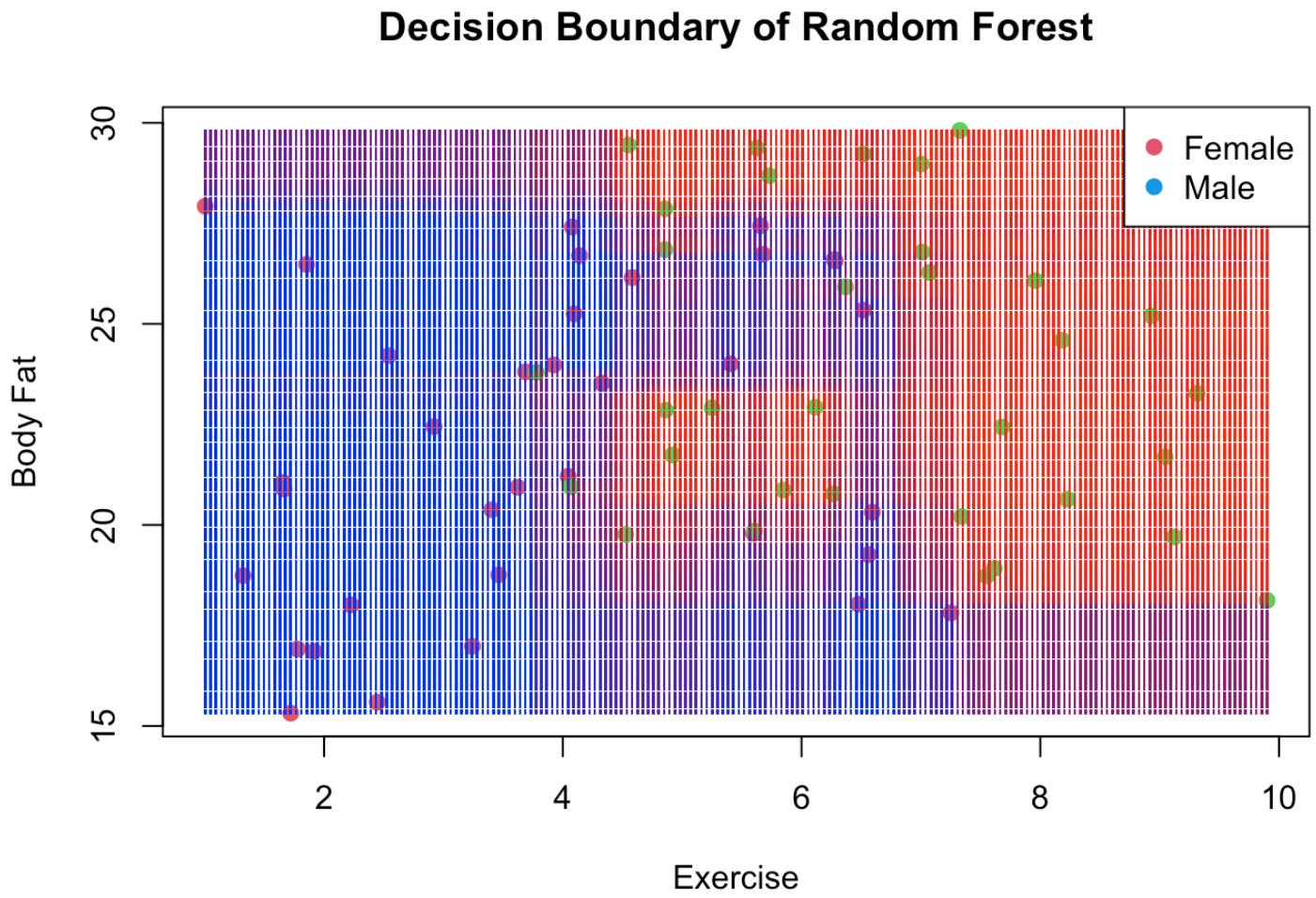
```
##
## predictions 0 1
##           0 8 7
##           1 7 8
```

- 7 instances were predicted as class 1 (Males) but were actually class 0 (False Positives).
- 7 instances were predicted as class 0 (Females) but were actually class 1 (False Negatives).

Random Forests - Example

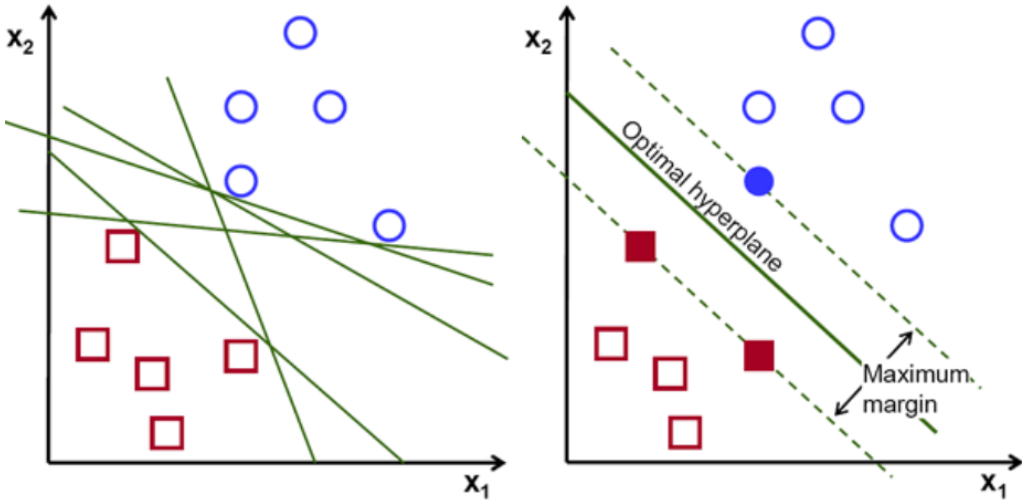


Random Forests - Example



Overview of SVMs

- In logistic regression, we learned about using a linear decision boundary to separate classes. This boundary is a hyperplane in the feature space that divides data points of different classes.
- However, when data is not linearly separable, a single hyperplane might not be sufficient to achieve accurate classification.
- SVMs step in to address this limitation by introducing the concept of "margin."

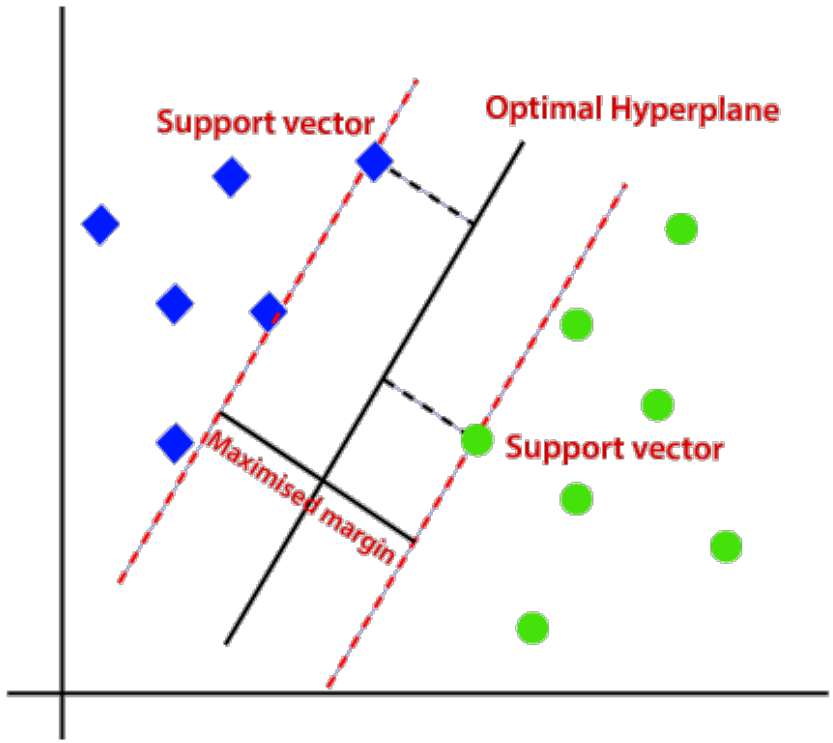


Overview of SVMs

- Support Vector Machines, or SVMs, are a class of supervised learning algorithms used for classification and regression tasks. They are particularly powerful when dealing with complex and high-dimensional data.
- While both logistic regression and SVM aim to find decision boundaries, an SVM seeks to find the best possible decision boundary (or hyperplane) that maximizes the margin between different classes of data points.
- This **margin** is the region between the hyperplane and the nearest data points of each class.
- The points that are closest to the hyperplane are called the **support vectors**. A separating line will be defined with the help of these data points.
- This distance is crucial because it ensures better generalization to new, unseen data.

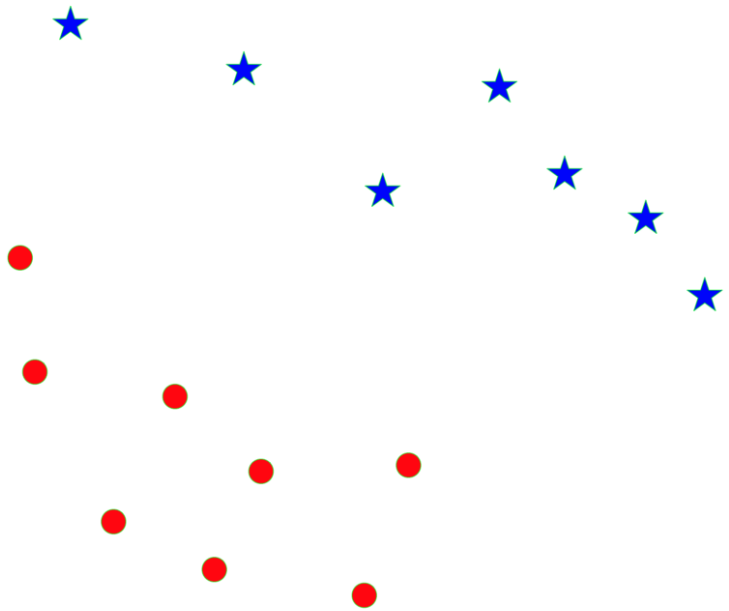
Overview of SVMs

- In a binary classification scenario, the separating hyperplane is the one that best separates the two classes.
- SVM is defined such that it is defined in terms of the support vectors only, we don't have to worry about other observations since the margin is made using the points which are closest to the hyperplane (support vectors)
  - whereas in logistic regression the classifier is defined over all the points.
  - Hence SVM enjoys some natural speed-ups.



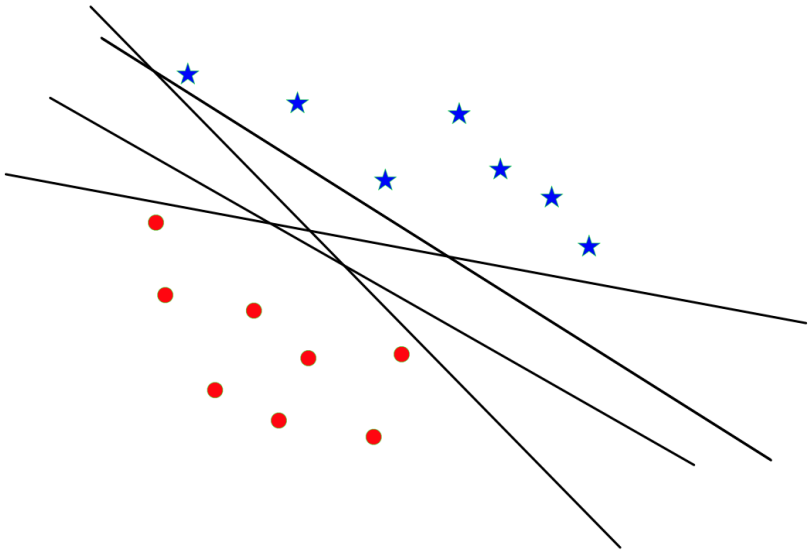
Overview of SVMs

- Let's imagine we have two tags: red and blue. We want a classifier that, given a pair of (x,y) coordinates, outputs a new data point as either red or blue.



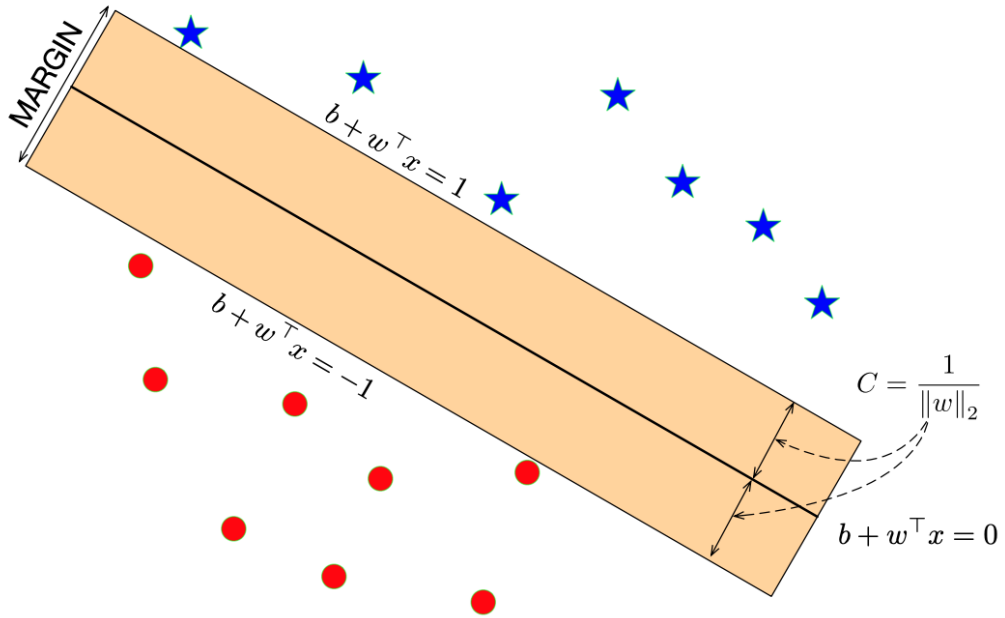
Overview of SVMs

- To classify these points, we can have many decision boundaries, but the question is which is the best and how do we find it?
- Since we are plotting the data points in a 2-dimensional graph we call this decision boundary a straight line but if we have more dimensions, we call this decision boundary a "hyperplane"



Overview of SVMs

- The best hyperplane is that plane that has the maximum distance from both the classes.
- This is done by finding different hyperplanes which classify the labels in the best way then it will choose the one which is farthest from the data points or the one which has a maximum margin.

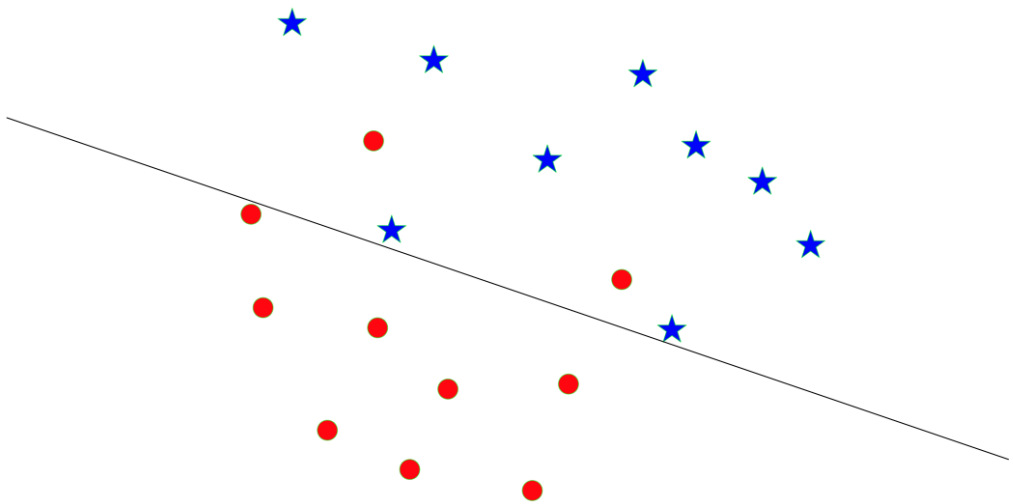


Mathematical intuition of SVMs

- Consider a binary classification problem with two classes, labeled as $+1$ and -1. We have a training dataset consisting of input feature vectors X and their corresponding class labels Y.
- The equation for the linear hyperplane can be written as: $w^T x + b = 0$
  - The vector W represents the normal vector to the hyperplane. i.e the direction perpendicular to the hyperplane.
  - The parameter b in the equation represents the offset or distance of the hyperplane from the origin along the normal vector w.
- Now, we want to pick w, b that maximizes the size of the margin (the region where we do not allow points to fall), while ensuring all points are correctly classified.

Mathematical intuition of SVMs

- How can we apply the max-margin principle if the data are not linearly separable?



Mathematical intuition of SVMs

- SVMs can be classified into hard margin and soft margin SVMs.
- Hard margin SVM aims for a strict separation between classes, but it might struggle with noisy data or outliers.
- Soft margin SVM introduces flexibility by allowing some data points to be misclassified to achieve a more robust decision boundary.

Mathematical intuition of SVMs

- Soft-margin SVM minimizes the average hinge losses plus the norm of the weights, where $z^{(i)} = w^T x^{(i)} + b$
- Hence, the soft-margin SVM can be seen as a linear classifier with hinge loss and an L_2 regularizer.
  - Hinge loss: $max(0, 1 - t^{(i)} z^{(i)})$
  - Penalty term: $\lambda \|w\|_2^2$

$$\min_{\mathbf{w},b} \sum_{i=1}^{N} \frac{1}{N} \max\left(0, 1 - t^{(i)} z^{(i)}\right) + \lambda \|\mathbf{w}\|_2^2$$

SVM - Example

- Remember the scenario we worked on in our logistic regression lecture.
- Let's apply an SVM classifier to the same data set we've seen for the logistic regression and the random forests classifier examples.
- Recall that the goal is to create a classification boundary to predict the gender (male or female) based on body fat percentage and exercise hours per week.
  - We're dealing with binary classification (two classes).
  - Male and female are our class labels.
- Again, we will divide the data into training set(70%) and testing set(30%).

```
# Glimpse the dataset
glimpse(data)
```

```
## Rows: 100
## Columns: 3
## $ exercise <dbl> 1.320818, 4.095401, 6.592474, 1.853295, 4.926636, 2.445720, 1…
## $ body_fat <dbl> 18.73851, 25.24797, 20.31670, 26.47923, 27.22607, 15.59223, 2…
## $ gender   <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
```

SVM - Example

- In R, **svm()** is the function we can use to build the SVM classifier.
- 'gender exercise + body_fat' is the formula we provide. This formula specifies the relationship we want to model.
- **kernel = "radial"**: The kernel parameter specifies the kernel function to be used. The "radial" kernel, also known as the Radial Basis Function (RBF) kernel, is commonly used for non-linear classification tasks. It's suitable for cases where the decision boundary is not linear.
- We can also specify the kernel to be **kernel = "linear"** if we know that our data is linearly separable.

```
# Train the SVM classifier
svm_model <- svm(gender ~ exercise + body_fat, data = data_train, kernel = "radial", probability = TRUE)
```

SVM - Example

- Again, we used the '**predict()**' function to test the performance of the trained SVM classifier on the testing set.
- Similar to previous examples, we computed the confusion matrix

```
# Make predictions on the test data
predictions <- predict(svm_model, data_test)

# Create a confusion matrix
conf_matrix <- table(predictions, data_test$gender)
print(conf_matrix)
```

```
##
## predictions  0  1
##           0  8  2
##           1  7 13
```

SVM - Example

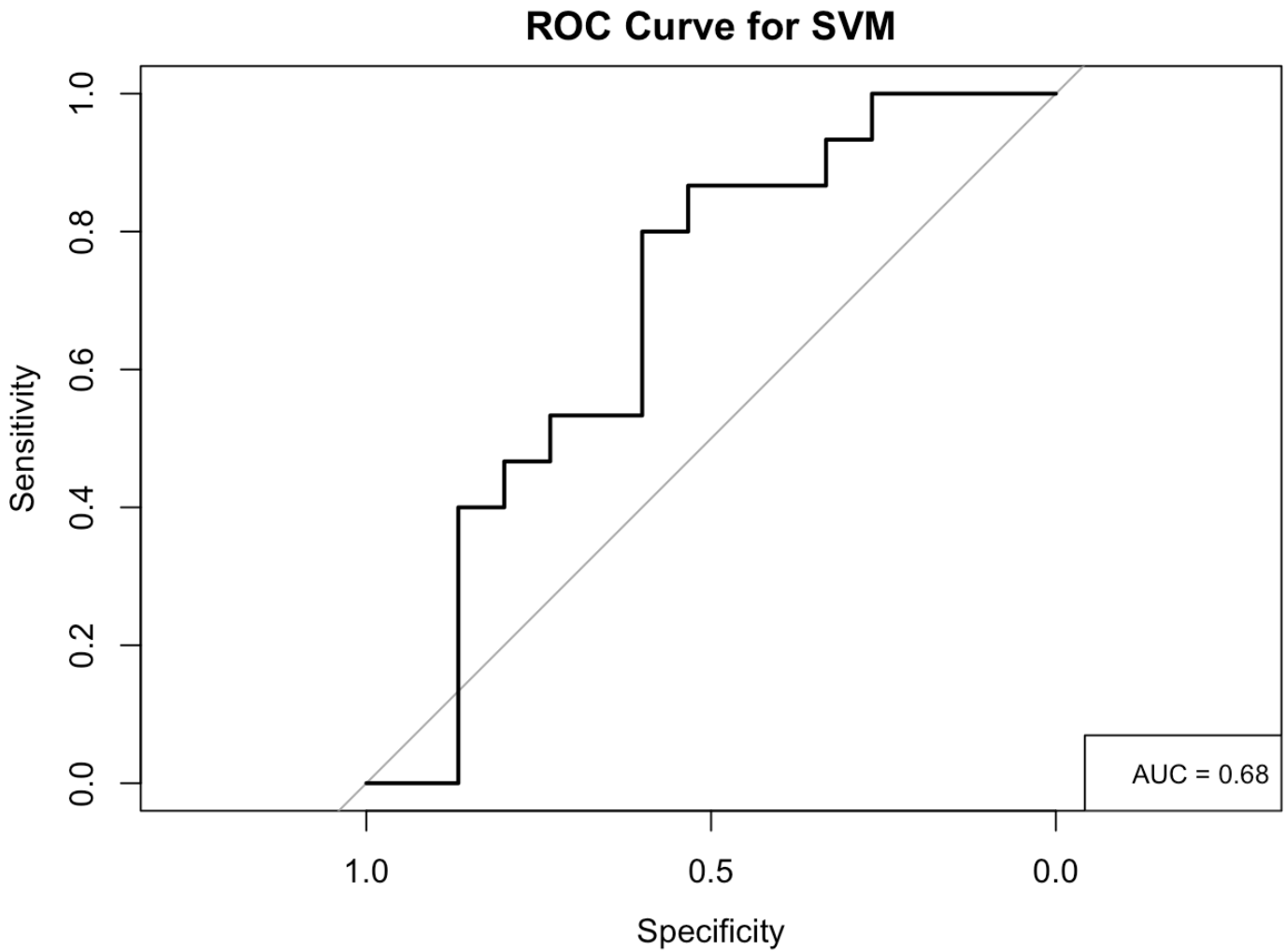
	Predicted 0	Predicted 1
Actual 0	True Negative	False Positive
Actual 1	False Negative	True Positive

Table: Confusion Matrix

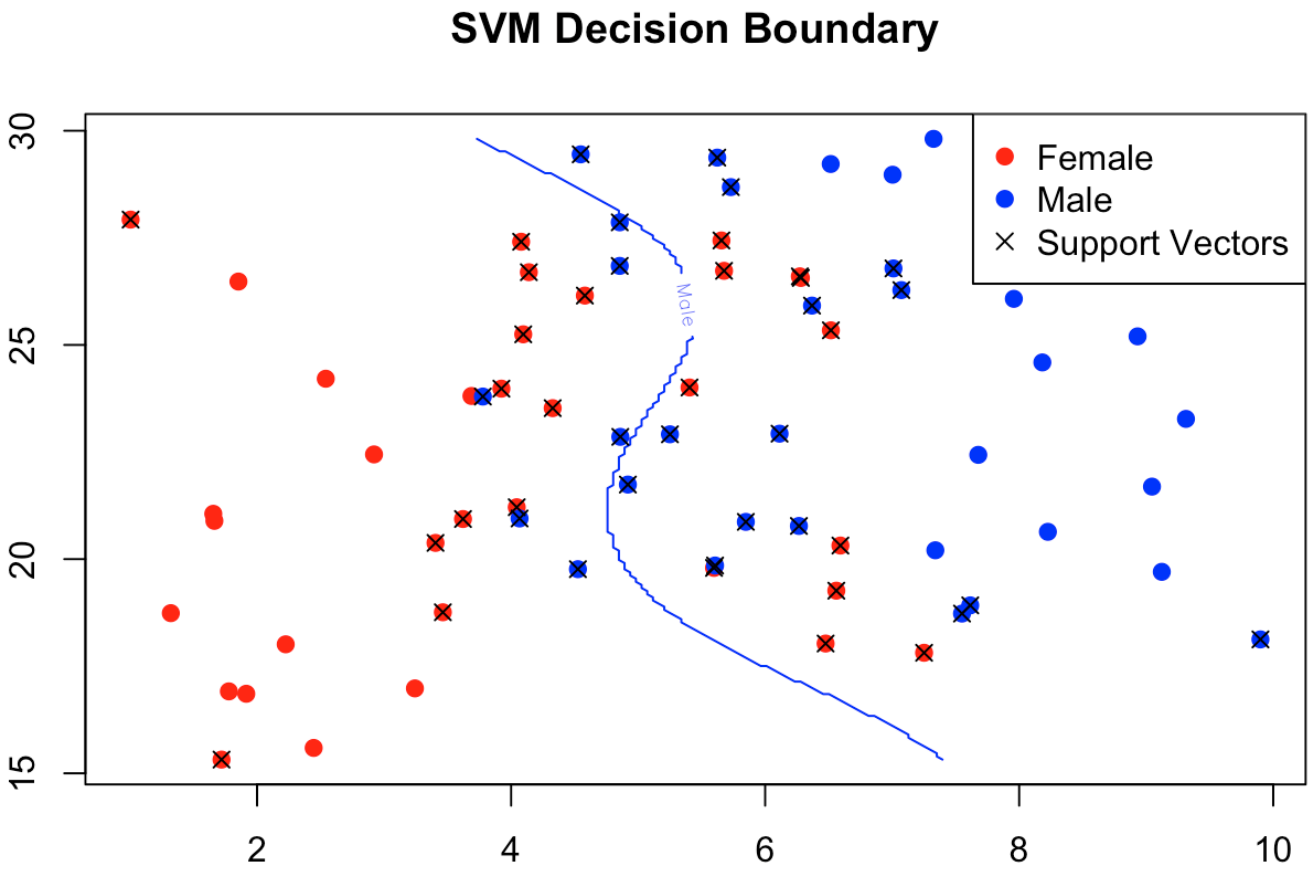
```
##
## predictions  0  1
##           0  8  2
##           1  7 13
```

- 2 instances were predicted as class 1 (Males) but were actually class 0 (False Positives).
- 7 instances were predicted as class 0 (Females) but were actually class 1 (False Negatives).

SVM - Example


ROC Curve for SVM
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity
AUC = 0.68

SVM - Example


SVM Decision Boundary
Female
Male
Support Vectors
Male
30
25
20
15
2
4
6
8
10

Decision Boundary

We've applied three classifiers to our data so far...

Decision Boundary

We've applied three classifiers to our data so far...

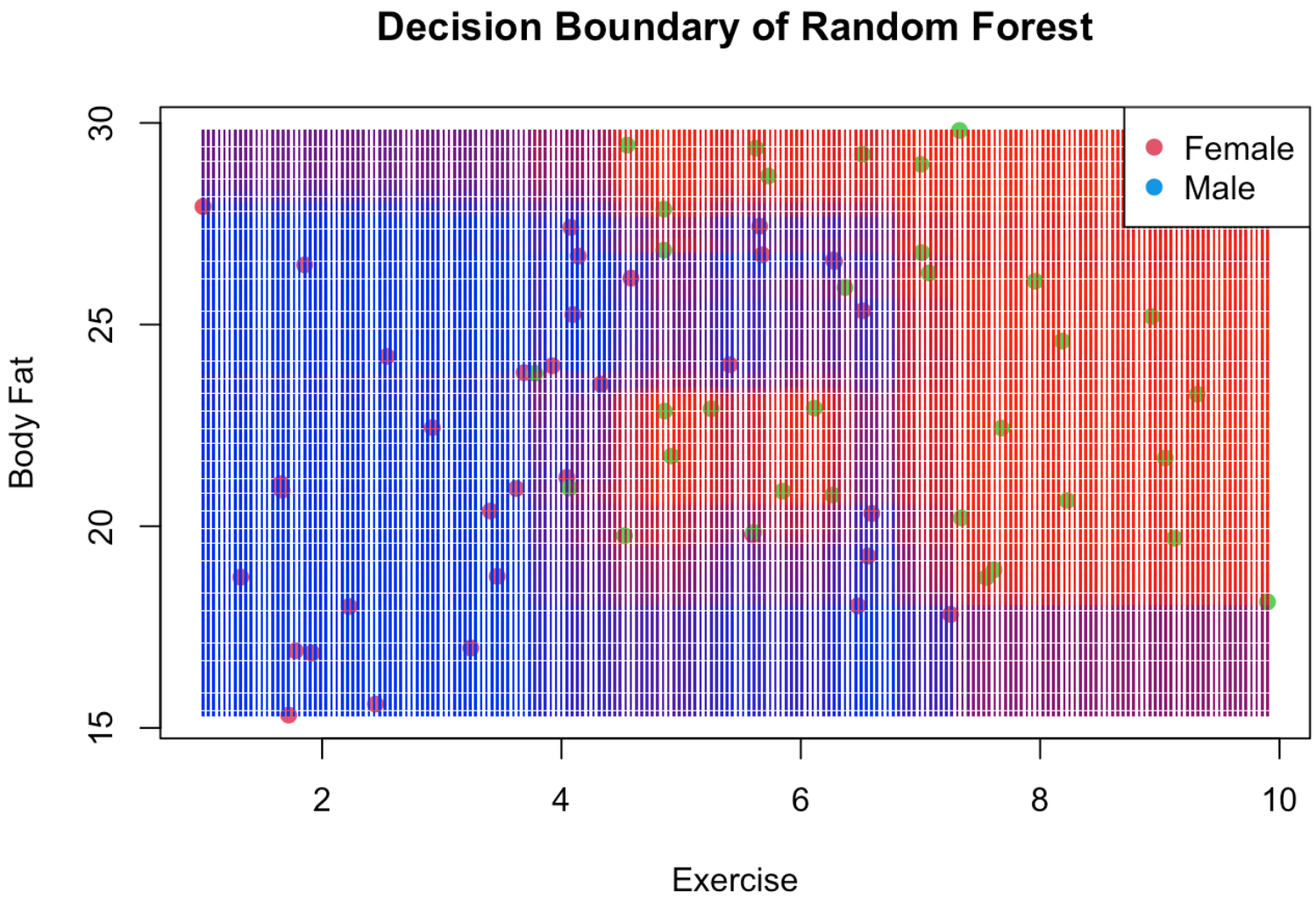
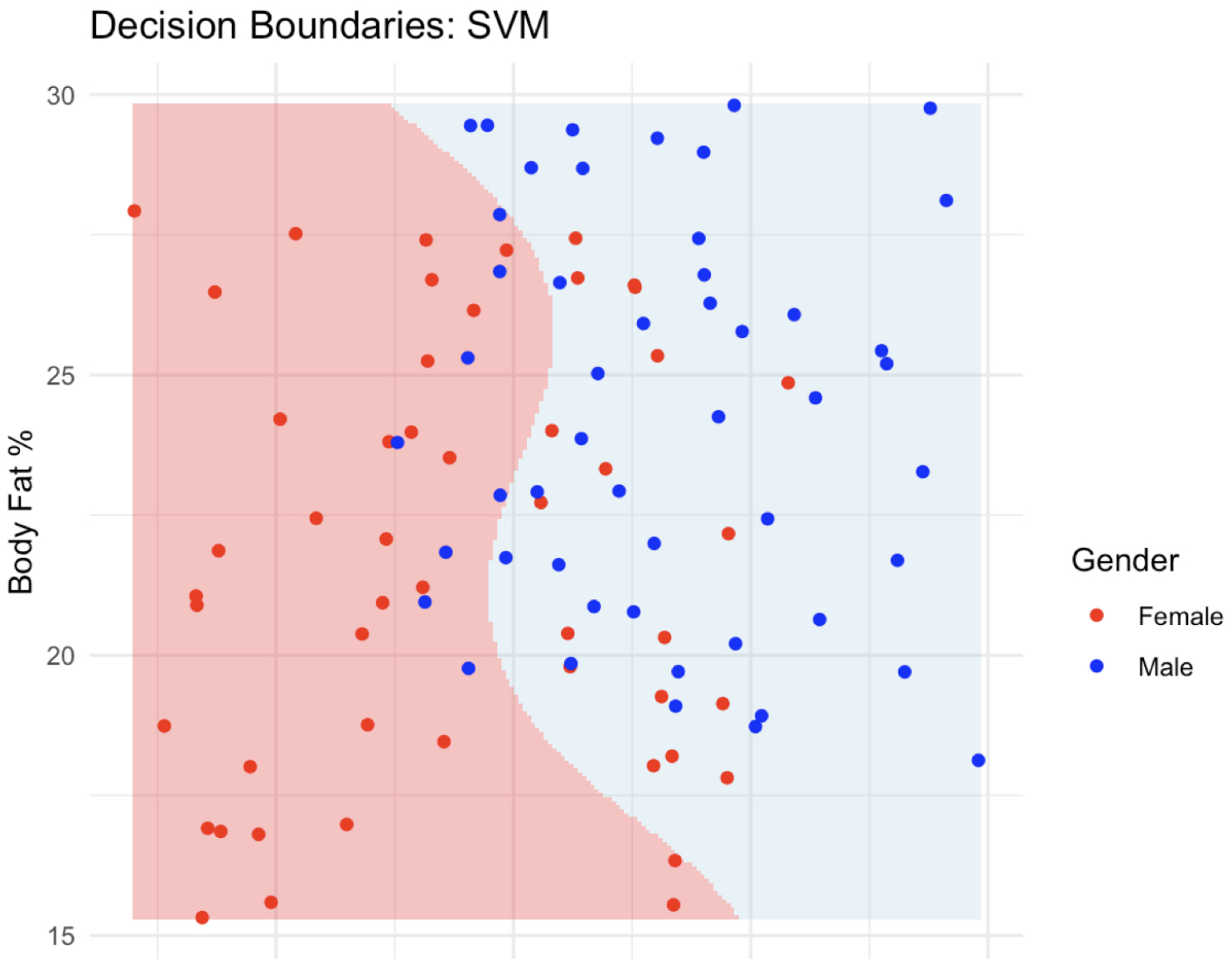


Figure: Random Forests

Decision Boundary

We've applied three classifiers to our data so far...



Performance metrics

Classifier	TN	FN	TP	FP
Logistic	7	8	12	3
Random Forests	8	7	8	7
SVM	8	7	13	2

Table: Confusion Matrix

Classifier	Accuracy	Sensitivity	Specificity	F1 score
Logistic	0.63	0.47	0.80	0.56
Random Forests	0.53	0.53	0.53	0.53
SVM	0.70	0.53	0.87	0.64

Table: Performance Metrics

Which classifier performs the best on our data?

Performance metrics

Classifier	TN	FN	TP	FP
Logistic	7	8	12	3
Random Forests	8	7	8	7
SVM	8	7	13	2

Table: Confusion Matrix

Classifier	Accuracy	Sensitivity	Specificity	F1 score
Logistic	0.63	0.47	0.80	0.56
Random Forests	0.53	0.53	0.53	0.53
SVM	0.70	0.53	0.87	0.64

Table: Performance Metrics

Which classifier performs the best on our data?

- SVM classifier appears to perform the best among the three, demonstrating higher accuracy, specificity, and F1 score.

Classifier Overview

- Logistic Regression:
  - Linear classifier.
  - Simple and interpretable.
  - Suitable for binary and multi-class classification.
  - Efficient for large datasets.
  - Assumes linear relationships.
- Random Forests:
  - Ensemble method (multiple decision trees).
  - Handles non-linear relationships.
  - Robust against overfitting.
  - Can handle high-dimensional data.
  - Provides feature importance.
- Support Vector Machines (SVM):
  - Margin-based classifier.
  - Effective in high-dimensional spaces.
  - Can capture complex non-linear relationships.
  - Robust against overfitting.
  - Provides clear decision boundaries.

Classifier Comparisons

- Non-Linearity:
  - Logistic Regression: Limited ability for non-linearity.
  - Random Forests: Highly capable for non-linearity.
  - SVM: Handles non-linearity with proper kernels.
- Performance on Complex Data:
  - Logistic Regression: Struggles with complex patterns.
  - Random Forests: Performs well on complex data.
  - SVM: Performs well with proper kernels.
- Overfitting:
  - Logistic Regression: Prone to overfitting with many features.
  - Random Forests: Less prone due to ensemble.
  - SVM: Can be controlled with regularization.
- Computation:
  - Logistic Regression: Efficient and quick to train.
  - Random Forests: Slower due to multiple trees.
  - SVM: Computationally intensive, especially for large datasets.